We Claim:

1. A tool for processing a p-code file, comprising:

analyzing said p-code file to identify those p-code methods within the file having associated with them at least one profile parameter above a threshold level; and

annotating said identified p-code methods in a manner adapted to enable preferential processing of said identified p-code methods by a compiler.

2. The tool of claim 1, wherein:

said p-code file comprises one of a Java class file, a C# file, an o-code file and a ground Java file.

3. The tool of claim 1, wherein:

said p-code file comprises a Java application file including Java classes, said Java class being annotated in a manner adapted to enable preferential processing of said identified Java classes by a Java virtual machine (VM) just-in-time (JIT) compiler.

4. The tool of claim 1, wherein said annotations are provided in-line with said identified p-code methods.

5. The tool of claim 1, wherein said annotations are provided as a separate file.

6. The tool of claim 1, wherein:

said at least one profile parameter comprises at least one of a method execution time, a frequency of method invocation, a number of instructions and a use of loop structures.

7. The tool of claim 1, wherein:

said at least one profile parameter comprises at least one of an execution time parameter, an input/output utilization parameter and a processor utilization parameter.

8. The tool of claim 1, wherein:

said analyzing comprises identifying at least one of a static profile parameter and a dynamic profile parameter.

9. The tool of claim 1, wherein:

said annotation comprises setting a normally unused bit within a method access flag field of an identified Java class file.

10. The tool of claim 1, wherein:

said annotation comprises selectively setting each of a plurality of normally unused bits within a method access flag field of an identified Java class file, wherein said unused bits are selectively set to define thereby a priority level of a respective annotated method.

11. The tool of claim 1, wherein:

each identified byte-code portion of said java application is associated with one of a plurality of priority levels, said annotation being indicative of respective priority levels.

12. The tool of claim 1, further comprising:

selectively pre-compiling at least a portion of said application file.

13. The tool of claim 12, wherein:

said precompiled portion of said application file is included within a virtual machine.

14. The tool of claim 1, wherein:

only a portion of said identified Java method is annotated in a manner adapted for subsequent compilation, said java method being annotated in a manner defining start and end byte code positions within said identified java method.

15. The tool of claim 1, wherein:

said Java application file comprises a ground java application file.

16. A method of adapting the interpretation of a p-code method by a virtual machine (VM), comprising:

compiling p-code methods associated with compilation indicative annotation; and

storing said compiled p-code methods in a cache for subsequent execution in place of corresponding interpreted p-code methods.

17. The method of claim 16, wherein:

said p-code methods are provided via one of a Java class file, a C# file, an o-code file and a ground Java file..

18. The method of claim 16, wherein:

said p-code file comprises a Java application file including Java classes and annotated Java classes, said annotated Java classes being preferentially compiled by a Java virtual machine (VM) just-in-time (JIT) compiler.

19. The method of claim 16, wherein said annotations are provided in-line with said identified p-code methods.

20. The method of claim 16, wherein said annotations are provided as a separate file.

21. The method of claim 16, further comprising:

in response to cache memory utilization above a threshold level, prioritizing the contents of said cache memory.

22. The method of claim 21, wherein:

said cache memory contents are prioritized by deleting from said cache compiled code associated with a least recently executed method.

23. The method of claim 21, wherein:

said cache memory contents are prioritized by deleting from said cache compiled code associated with a previously compiled method having a lower priority level than a presently compiled method.

24. The method of claim 20, wherein:

compiled byte-code stored in said cache is accessed via a cache map, said cache map being updated in response to a change in cache utilization.

25. The method of claim 20, further comprising:

compiling non-annotated byte-code within said Java application if said non-annotated byte-code utilizes VM resources beyond a threshold level.

26. The method of claim 25, wherein:

said compiled non-annotated byte-code is assigned a priority level in accordance with said utilized VM resources.

27. The method of claim 26, wherein:

said priority level of said annotated byte-code is further adapted in accordance with said utilized VM resources.

28. The method of claim 20, further comprising:

said compiled annotated byte-code is assigned a priority level in accordance with said utilized VM resources.

29. The method of claim 28, wherein:

said priority level of said annotated byte-code is further adapted in accordance with said utilized VM resources.

30. The method of claim 26, wherein said VM resources comprise at least one of an execution time parameter, an input/output utilization parameter and a processor utilization parameter.